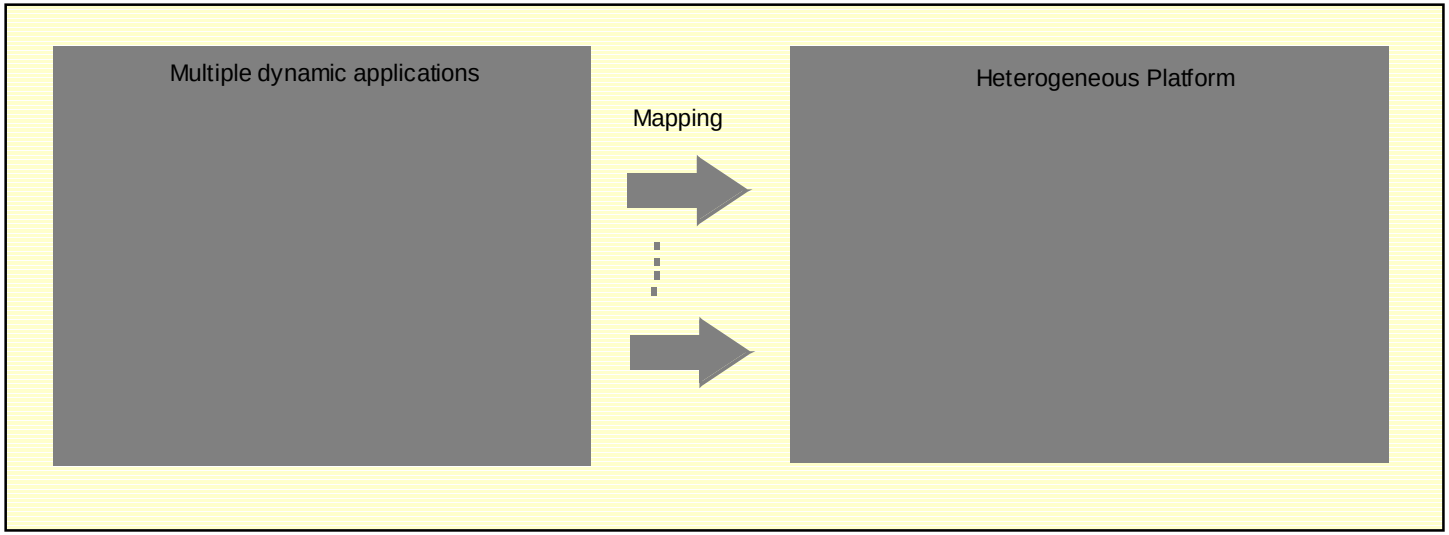
Real-Time Adaptive Resource Management

Sunondo Ghosh
PI's: Rakesh Jha, Walt Heimerdinger
Honeywell Technology Center
{sghosh, jha, walt}@htc.honeywell.com
612-951-7513 (phone), 612-951-7438 (fax)

Problem Statement

Future embedded systems (SC-21, JSTARS, AWACS) will be characterized by dynamic variability in resource demands and availability:



Applications may be serial, parallel, or distributed, each with specific performance requirements. Resource needs may vary during execution due to data-dependence, changes in external environment etc.

Goal

Develop techniques for continual adaptive resource (re)allocation in response to a variety of dynamic triggers -- detected performance shortfall; application arrival, departure; direct request by applications or users, etc.

Resource Management Model

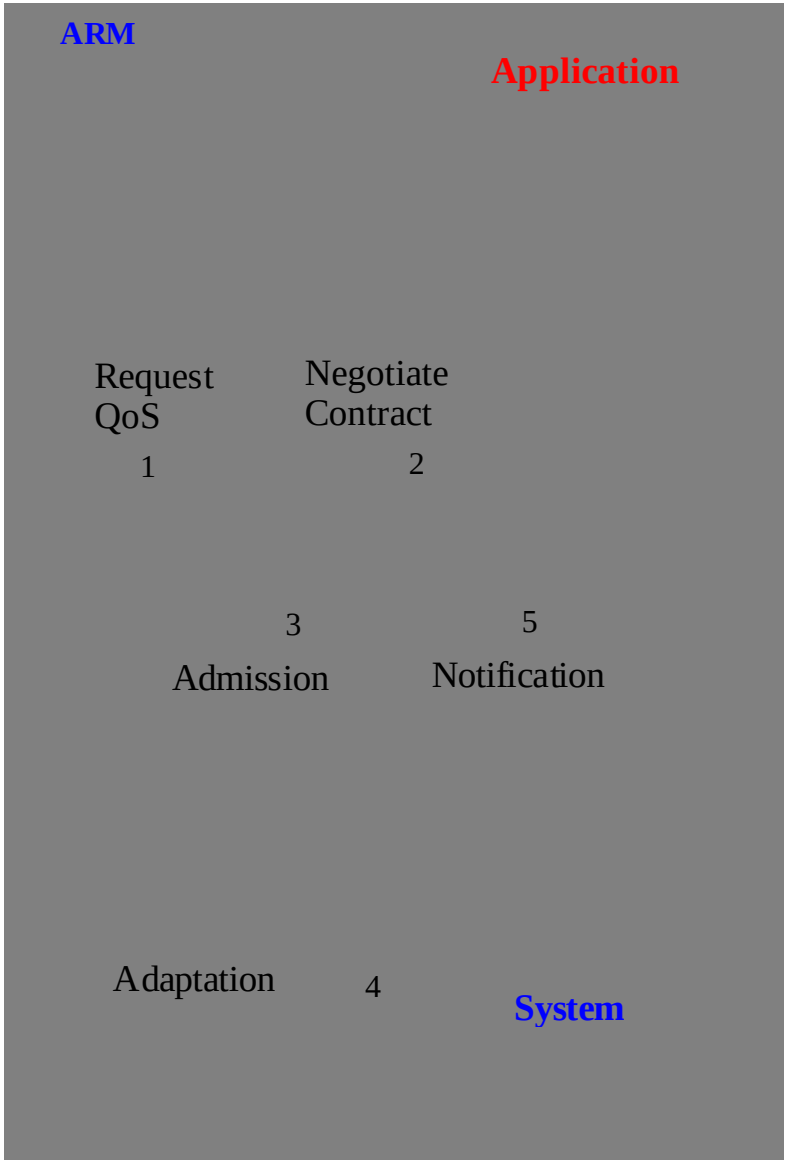
Present resources to applications as a pool dynamically customizable to their needs

Based on QoS contracts and dynamic adaptation within contract.

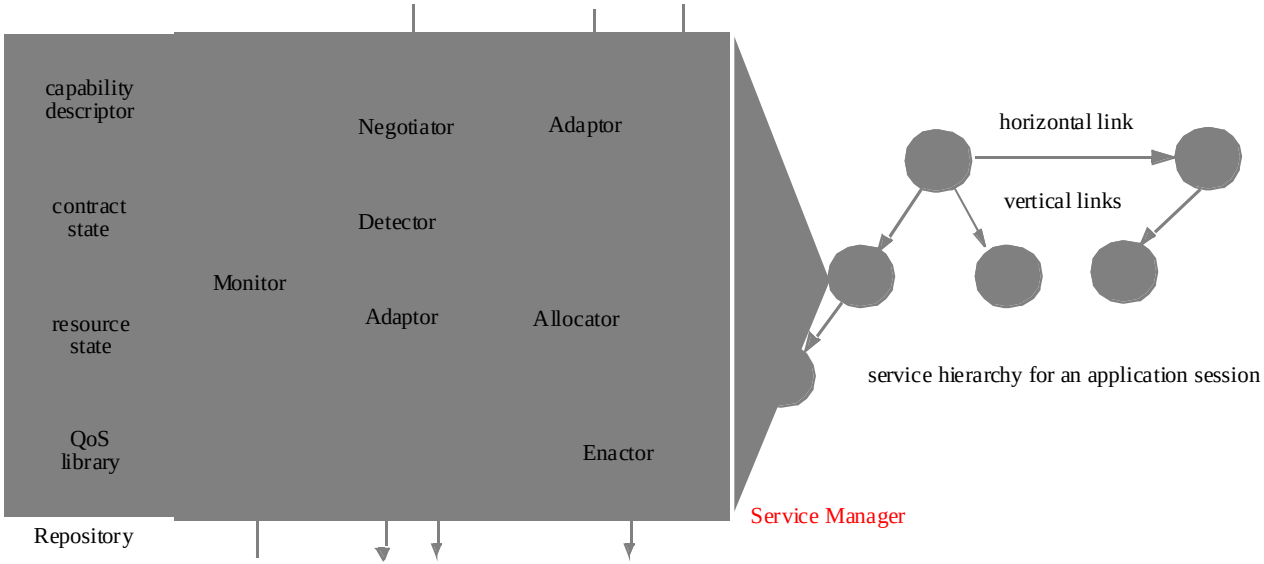
Adaptations triggers caused by changes in resource demands and availability

- **Arrival/departure of applications**
- **Direct request by applications/users**
- **QoS deviations detected by ARM**

Adaptations: QoS expansion, QoS reduction, reconfiguration and tuning.



RTARM Architecture



ARM achieved by a hierarchy of service managers

Vertical links = service composition

Horizontal links = application flow, or precedence, or QoS dependencies

Negotiator negotiates contract for service

Allocator determines QoS feasibility

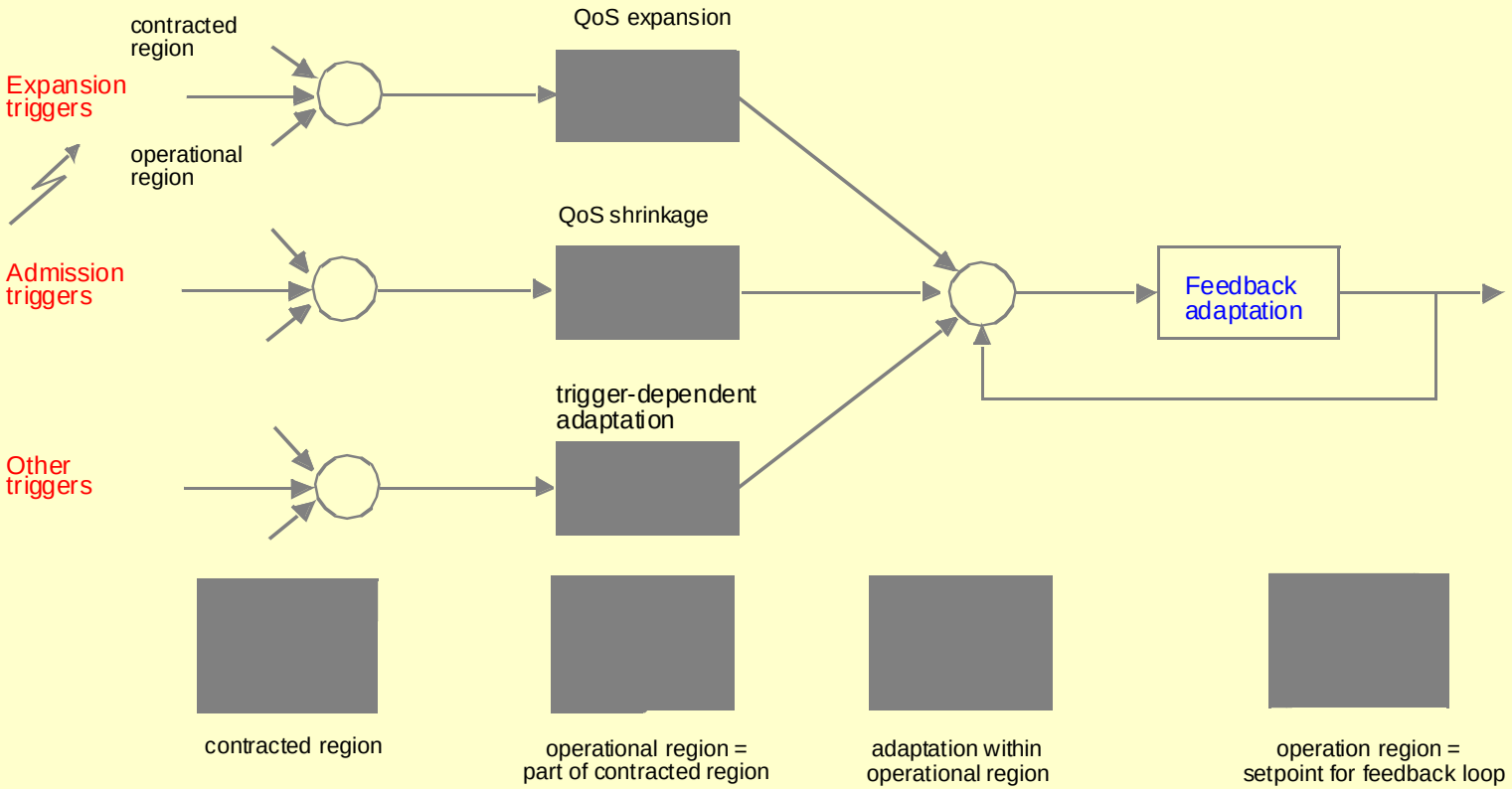
Monitor monitors delivered QoS

Detector detects QoS deviations

Adaptors selects adaptive response

Enactor effects (re)allocations

Two-Stage Adaptation Structure



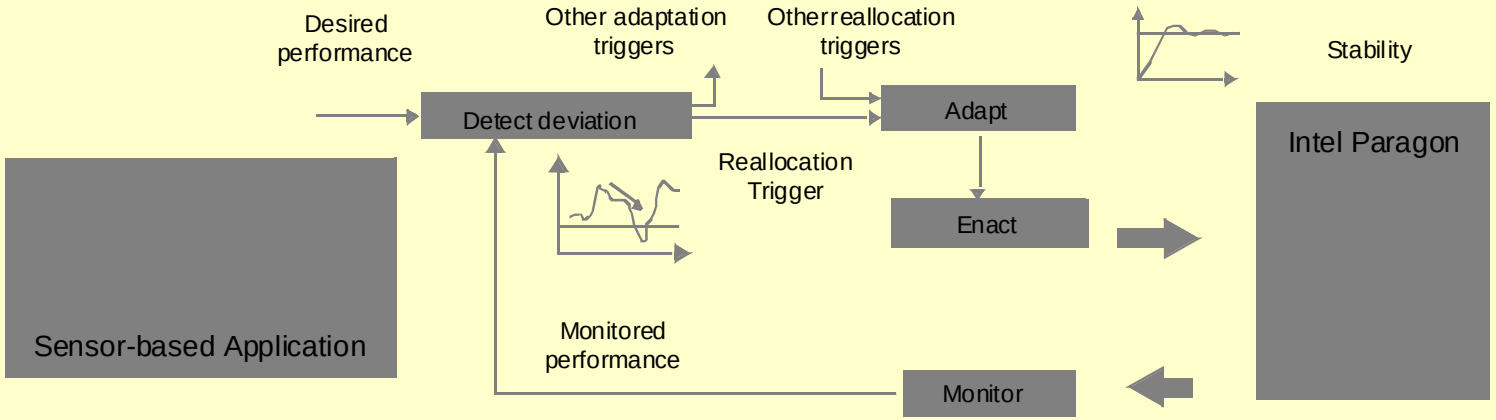
Adaptation across applications

QoS shrinkage, QoS expansion, preemption

Adaptation within applications

Application reconfiguration, redistribution of resources across components

Results: Feedback Adaptation



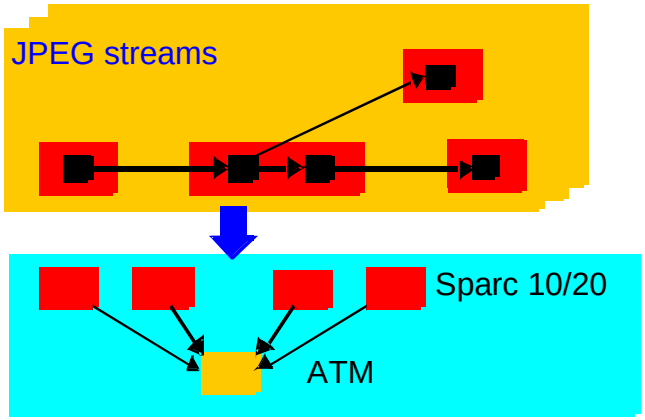
Continual resource reallocation to maintain throughput despite changing workload

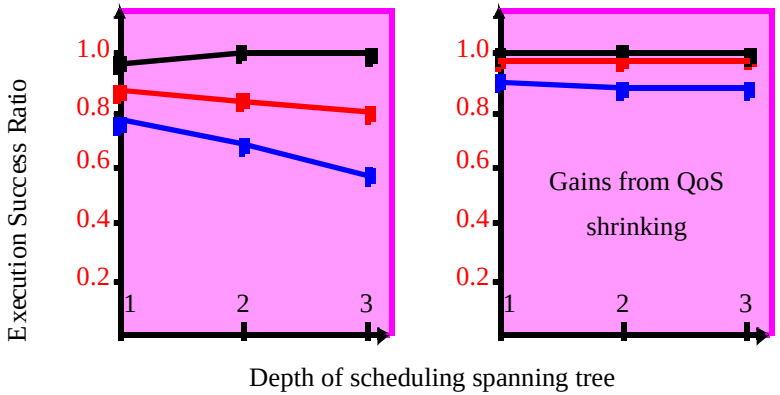
Best-effort service for sensor-based multi-pipeline applications

Greedy processor reallocation, then assignment based on a) cascading, b) incremental branch and bound algorithm

6-stage multi-pipeline on a 16-node paragon on an ATR-based application

Results: QoS-Based Admission + Adaptation





Decentralized negotiation and adaptation for multimedia stream applications

Guarantee critical apps at QoSmin, maximize # apps, try to achieve QoSmax for all apps

Negotiation protocol overhead: 20 ms for a negotiation spanning tree depth of 3; similar overhead for TCP/ATM, ATM/AAL5

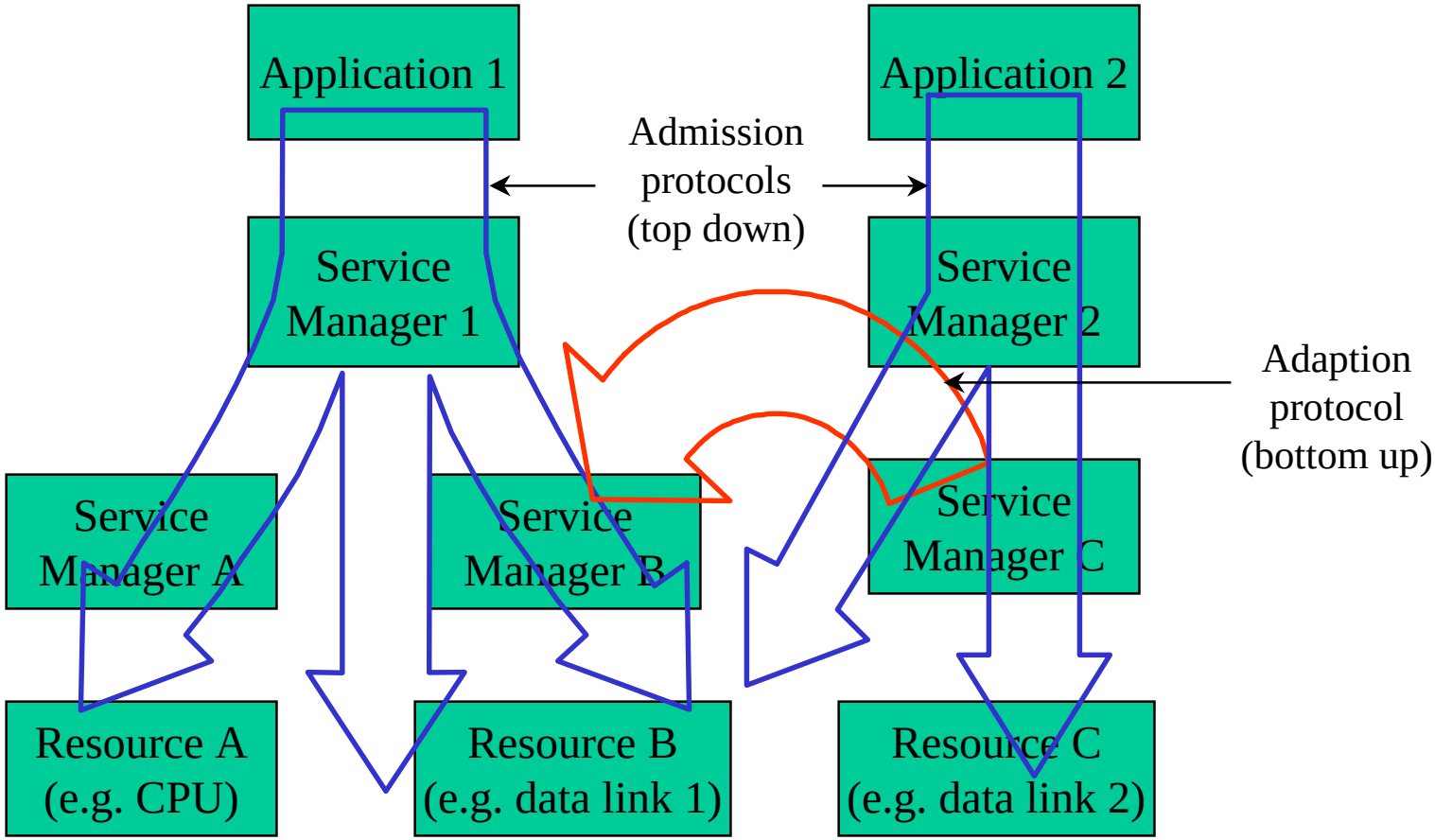
Workload: 1 per sec. arrival of 30fps streams

Significant performance gains achieved from QoS shrinking and QoS expansion

Steps for fault tolerance

- Fault detection triggers
  - Acceptance test in application
  - Self test in system (built-in test)
  - Timeout mechanism if system is fail silent
- Adaptation mechanisms
  - rerun adaptation protocol at successively higher levels
  - reallocate (or invalidate contract in necessary)
  - application can change request for quality if contract invalidated

Adaptation Information Flows for Admission and Fault Adaptation



Fault tolerance in RT-ARM

- Current triggers
  - application arrival, departure, or increased request for resources
  - top down
- New trigger: fault
  - bottom up
- Effect of new trigger
  - Either reallocation of contract to new resource
  - Or invalidation of contract
  - Application is informed

Plan for work with JPL

- Demonstrate adaptation infrastructure with potential for fault tolerance
  - Host on JPL testbed (current prototype on Solaris/NT & ATM networks)
  - Respond to testbed-triggered faults
- Jointly develop QoS-aware applications
  - Integrate QoS-aware applicatio with adaptation infrastructure

What we need from JPL

- Fault detection mechanisms
- QoS aware applications
- Information about current applications
- Access to JPL testbed
- Funding

Timeframe

- Developing QoS aware applications will take time
- Adding fault tolerance to RT-ARM can be done in parallel

Other issues

- Execution: To be decided, preferably over internet
- What will be left behind: Nothing at the present time, adaptation infrastructure and algorithms in the future